

(19) 日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11) 特許出願公開番号

特開平10-191090

(43) 公開日 平成10年(1998) 7月21日

(51) Int.Cl.⁶

識別記号

F I

H 0 4 N 1/60
B 4 1 J 2/525
G 0 6 T 1/00
G 0 9 G 5/06
H 0 4 N 1/46

H 0 4 N 1/40 D
G 0 9 G 5/06
B 4 1 J 3/00 B
G 0 6 F 15/66 N

3 1 0

審査請求 未請求 請求項の数10 O L (全 14 頁) 最終頁に続く

(21) 出願番号 特願平9-274697

(22) 出願日 平成9年(1997)10月7日

(31) 優先権主張番号 特願平8-282726

(32) 優先日 平8(1996)10月24日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000002369

セイコーエプソン株式会社

東京都新宿区西新宿2丁目4番1号

(72) 発明者 深沢 賢二

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

(72) 発明者 笠原 広和

長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

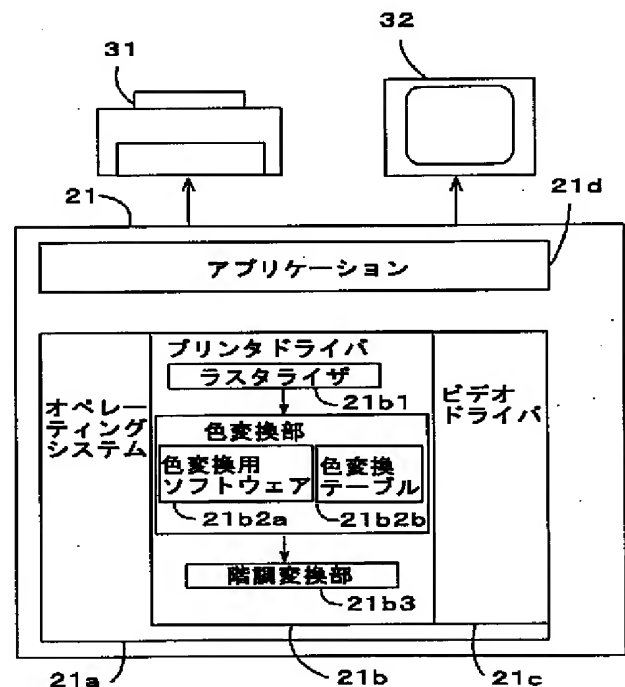
(74) 代理人 弁理士 鈴木 喜三郎 (外2名)

(54) 【発明の名称】 色変換テーブルの製造装置及び製造方法並びに記録媒体

(57) 【要約】 (修正有)

【課題】 プリンタドライバで使用する色変換テーブルはユーザの環境によっては最適なものとは限らなかった。

【解決手段】 画像処理装置を構成するコンピュータ21にてインストーラが実行されると、元色変換テーブルから色変換テーブル21b2bを生成するが、このときの格子点増加処理では、ラグランジュの補間公式を利用した非線形補間演算で格子点を増加させたり、線形補間で格子点を増加させるなどし、また、その際に固定した格子点の数であっても良いし環境や入力画像に応じた格子点の数としてもよく、小さなサイズの元色変換テーブルから適切なサイズの色変換テーブル21b2bを生成することができる。



【特許請求の範囲】

【請求項1】 異なる表色空間の間で階調表色データを変換するために変換元の表色空間に複数の格子点を設定し、この格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルを生成する色変換テーブルの製造装置であって、
少数の格子点において変換の対応関係を記憶する元色変換テーブルと、

この元色変換テーブルの格子点を補間演算によって増加させて変換に利用する色変換テーブルを生成する補間手段とを具備することを特徴とする色変換テーブルの製造装置。

【請求項2】 上記請求項1に記載の色変換テーブルの製造装置において、上記補間手段は、複数の格子点の対応関係から非線形補間演算で補間する非線形補間演算手段を具備することを特徴とする色変換テーブルの製造装置。

【請求項3】 上記請求項2に記載の色変換テーブルの製造装置において、上記元色変換テーブルは、均等な間隔の格子点となっていることを特徴とする色変換テーブルの製造装置。

【請求項4】 上記請求項1に記載の色変換テーブルの製造装置において、上記補間手段は、複数の格子点の対応関係から線形補間演算で補間する線形補間演算手段を具備することを特徴とする色変換テーブルの製造装置。

【請求項5】 上記請求項1～請求項4に記載の色変換テーブルの製造装置において、上記補間手段は、補間で増加させる格子点の数を選択可能としていることを特徴とする色変換テーブルの製造装置。

【請求項6】 上記請求項5に記載の色変換テーブルの製造装置において、上記補間手段は、増加させる格子点の数を環境に応じて設定することを特徴とする色変換テーブルの製造装置。

【請求項7】 上記請求項5に記載の色変換テーブルの製造装置において、上記補間手段は、増加させる格子点の数を変換画像の種類に応じて設定することを特徴とする色変換テーブルの製造装置。

【請求項8】 上記請求項1に記載の色変換テーブルの製造装置において、上記色変換テーブルはコンピュータにて参照するとともに、通常時は同コンピュータの補助記憶装置に上記元色変換テーブルを記憶するとともに、同色変換テーブルの参照実行時に上記コンピュータの主記憶領域に展開することを特徴とする色変換テーブルの製造装置。

【請求項9】 異なる表色空間の間で階調表色データを変換するために変換元の表色空間に複数の格子点を設定し、この格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルを生成する色変換テーブルの製造方法であって、
少数の格子点において変換の対応関係を記憶する元色変

換テーブルの前記格子点を補間演算によって増加させて変換に利用する色変換テーブルを生成することを特徴とする色変換テーブルの製造方法。

【請求項10】 異なる表色空間の間で階調表色データを変換するために変換元の表色空間に複数の格子点を設定し、この格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルをコンピュータにて生成する色変換テーブル作成プログラムを記録した記録媒体であって、

10 少数の格子点において変換の対応関係を記憶する元色変換テーブルの前記格子点を補間演算によって増加させて変換に利用する色変換テーブルを生成することを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、異なる表色空間の間で階調表色データを変換するために変換元の表色空間での格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルを生成する色変換テーブルの製造装置および製造方法並びに色変換装置に関する。

【0002】

【従来の技術】従来、この種の色変換テーブルとして、コンピュータ上のカラー画像をカラー印刷するカラー印刷システムが知られている。

【0003】コンピュータの内部では、カラー画像は縦横に並べられた各画素ごとについて赤緑青の三原色（R，G，B）で階調表示されているが、一般のカラー印刷装置においてはシアン、マゼンダ、イエローの三色（C，M，Y）あるいはこれにブラックを加えた（C，M，Y，K）四色で階調表示のない状態で印刷される。従って、カラー印刷するためには赤緑青の三原色（R，G，B）の表示からシアン、マゼンダ、イエローの三色（C，M，Y）の表示への変換の作業と、階調表示から階調のない表示への階調変換の作業が必要となる。なお、色空間自体は一つの空間であるものの、座標の取り方によって表示が異ならざるをえないため、以下においては、便宜上、座標の取り方に応じた表色空間と呼ぶことにする。

【0004】この（R，G，B）表示から（C，M，Y）表示への変換は変換式によって一義的に定まるものではなく、それぞれの階調を座標とする色空間について相互に対応関係を求めておき、この対応関係から逐次変換するのが通常である。ここにおいて、少なくとも変換元の（R，G，B）表示が各色について256階調であったとすれば、約1670万個（ $256 \times 256 \times 256$ ）の要素の色変換テーブルを持たなければならない。

【0005】効率的な記憶資源の利用を考えた結果、すべての座標値についての対応関係を用意しておくのではなく、適当なとびとびの格子点について対応関係を用意

しておき、補間演算を併用するようにしている。すなわち、(R, G, B)表色空間の中のある座標の色について(C, M, Y)表色空間の対応関係を求めるときには同座標を取り囲む格子点の対応関係を利用し、線形補間などの演算を経て同座標の対応関係を求めている。

【0006】このような色変換テーブルは、一般にプリンタドライバが備えており、色変換テーブルを含めたプリンタドライバ自体は個々のカラー印刷装置に対応して一つだけが提供されている。従って、色変換テーブルについても、記憶資源との対比から適当に定められた格子点の数に特定されていた。

【0007】

【発明が解決しようとする課題】上述した従来の色変換テーブルにおいては、プリンタドライバを提供する側が一般的な記憶資源との対比に基づいて色変換テーブルを作成しているため、必ずしもユーザの環境によっては最適なものとは限らないという課題があった。すなわち、ユーザの環境によってはまだまだ大きいという場合もあるし、より大きなサイズとした方が良いという場合もあった。

【0008】さらに、色変換テーブルの大きさによって印刷品質にも差が生じるため、一定の大きさの色変換テーブルでは十分ではないという課題もあった。

【0009】本発明は、上記課題にかんがみてなされたもので、ユーザの環境などに最適な色変換テーブルを生成することが可能な色変換テーブルの製造装置および製造方法の提供を目的とする。

【0010】

【課題を解決するための手段】上記目的を達成するため、請求項1にかかる発明は、異なる表色空間の間で階調表色データを変換するために変換元の表色空間での格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルを生成する色変換テーブルの製造装置であって、少数の格子点において変換の対応関係を記憶する元色変換テーブルと、この元色変換テーブルの格子点を補間演算によって増加させて変換に利用する色変換テーブルを生成する補間手段とを具備する構成としてある。

【0011】上記のように構成した請求項1にかかる発明においては、もともと存在しているのが格子点を少数とした元色変換テーブルであり、記憶資源をわずかしかなければ必要としない。そして、補間手段はこの元色変換テーブルの格子点を補間演算によって増加させて色変換テーブルを生成し、この生成された色変換テーブルを色変換に利用する。

【0012】元色変換テーブルにおける格子点の数は、生成される色変換テーブルと比較して相対的に少なければよく、必ずしも極めて少ない数値である必要はない。特に、三次元の格子点であれば、格子点の多少は3乗で変化してくる。すなわち、格子点を半分にすることで色

変換テーブルのサイズは1/8の大きさとなり、少しの差でも記憶資源に対して大きな効果を与えることになる。

【0013】補間演算手段については各種の演算を適用することが可能であり、その一例として、請求項2にかかる発明は、請求項1に記載の色変換テーブルの製造装置において、上記補間手段は、複数の格子点の対応関係から非線形補間演算で補間する非線形補間演算手段を具備する構成としてある。

10 【0014】上記のように構成した請求項2にかかる発明においては、補間手段の非線形補間演算手段が複数の格子点の対応関係から非線形補間演算で格子点を補間する。

【0015】非線形補間演算を行うことにより、増加される格子点での対応関係が正確に再現されることになる。従って、元色変換テーブルの格子点を少なくとも非常に再現性の良い結果を得ることが可能となる。

20 【0016】一方、元色変換テーブルの格子点の間隔についてはこの補間演算にも関連があり、その一例として、請求項3にかかる発明は、請求項2に記載の色変換テーブルの製造装置において、上記元色変換テーブルは、均等な間隔の格子点となった構成としてある。

【0017】非線形補間演算においては必ずしも格子点間隔が均等である必要はないが、不均等な格子点間隔であるとする演算式の係数が複雑になってくる。そして、少なくとも三次元の立方体内で補間演算で格子点を増加させようとする場合には、各軸方向毎に複数の格子点から中間的な格子点を求めるような作業が必要になり、このような場合に演算式の係数が複雑になってくると補間演算が煩雑になり、作業が煩わしい。これに対して均等な格子点間隔である場合には一部の係数などが一定となってループ処理も適用しやすくなるなどの状況が生じる。

【0018】むしろ、非線形補間演算以外の他の補間演算によっては均等である方が好ましい場合もあるし、不均等である方が好ましい場合もあり、補間演算に応じて適宜変更可能である。

40 【0019】また、非線形補間演算以外の演算を利用する一例として、請求項4にかかる発明は、請求項1に記載の色変換テーブルの製造装置において、上記補間手段は、複数の格子点の対応関係から線形補間演算で補間する線形補間演算手段を具備する構成としてある。

【0020】線形補間の場合には演算式自体が複雑でないというメリットもあるし、演算に必要な格子点が軸方向に対して二点であるという性質がある。従って、対応関係が大きく変化する部分においては格子点を細かくすることにより、正確な対応関係を容易に得られるようになるし、逆に、対応関係がさほど変化しない部分においては格子点を粗くするといったことも容易になる。

50 【0021】補間演算手段が増加させる格子点の数は、

必ずしも一定である必要はなく、その一例として、請求項5にかかる発明は、請求項1～請求項4に記載の色変換テーブルの製造装置において、上記補間手段は、補間で増加させる格子点の数を選択可能とした構成としてある。

【0022】増加される格子点の数により色変換テーブルのサイズが変化するし、補間演算によっては格子点の色変換での変換精度に影響を与える場合もある。従って、補間で増加させる格子点の数が選択可能であることにより、ユーザの環境に対して最適な格子点の数とすることが可能となってくる。

【0023】この場合において、請求項6にかかる発明は、請求項5に記載の色変換テーブルの製造装置において、上記補間手段は、増加させる格子点の数を環境に応じて設定する構成としてある。

【0024】格子点の数によって色変換テーブルのサイズが変化するし、色変換時のヒット率も変わってくる。さらには、増加させる格子点の位置によっては補間演算を容易にさせたりすることにもなる。従って、環境に応じて総合的に格子点を増加させることにより、最適な色変換を提供できる。環境に応じて増加する格子点を選択する方針として、記憶資源が多ければ大きなサイズの色変換テーブルとすればよいし、格子点が多いほど色変換精度が高い場合において色変換精度を高くすることを望むのであれば格子点を多くするようにすればよいし、補間演算時に2の累乗での乗除算が容易であるようならば、格子点の間隔が2の累乗となるようにすればよい。

【0025】さらに、同様に増加する格子点を選択するにあたり、請求項7にかかる発明は、請求項5に記載の色変換テーブルの製造装置において、上記補間手段は、増加させる格子点の数を変換画像の種類に応じて設定する構成としてある。

【0026】上述したように、格子点が多いほど色変換精度が高い場合があり、そのような場合において画像によっては色変換精度を高く要求するものもあればさほど必要としないようなものもある。このため、変換画像の情報が色変換精度を高く要求するようなものであれば格子点をできるだけ多くし、色変換精度をさほど要求しないようなものであれば格子点をさほど多くしない。この場合の方針として、例えば、オペレーティングシステムなどから変換画像の種類を知ることができるような場合に、ファイルの拡張子がビットマップであれば写真などの色情報の重要度が高いものと判断して格子点をできる限り多くする一方、ファイルの拡張子がドローデータやビジネスグラフを指すような場合には色情報の重要度は低いものと判断して格子点をさほど増やさないといったことが有効である。

【0027】発明の思想の具現化例における他の一例として、請求項9にかかる発明は、異なる表色空間の間で階調表色データを変換するために変換元の表色空間に複

数の格子点を設定し、この格子点に変換先の表色空間での階調表色データを対応させた色変換テーブルを生成する色変換テーブルの製造方法であって、少数の格子点において変換の対応関係を記憶する元色変換テーブルの前記格子点を補間演算によって増加させて変換に利用する色変換テーブルを生成することを特徴としておる。

【0028】すなわち、必ずしも実体のある装置に限らず、その方法としても有効であることに相違はない。

【0029】ところで、このような色変換テーブルを備える色変換装置は単独で存在する場合もあるし、ある機器に組み込まれた状態で利用されることもあるなど、発明の思想としてはこれに限らず、各種の態様を含むものである。従って、ソフトウェアであったりハードウェアであったりするなど、適宜、変更可能である。

【0030】その一例として、印刷インクに対応した表色空間に対して異なる表色空間の階調表色データを変換するにあたり、少数の格子点において変換の対応関係を記憶する元色変換テーブルを使用し、格子点を補間演算によって増加させて色変換テーブルを生成し、この生成した色変換テーブルを利用して色変換する構成とすることもできる。

【0031】すなわち、プリンタドライバは印刷インクに対応した表色空間に対して異なる表色空間の階調表色データを変換するために、色変換テーブルを参照することになるが、この際に少数の格子点の元色変換テーブルから補間して格子点を増し、格子点を増加した色変換テーブルを使用して色変換する。

【0032】発明の思想の具現化例として色変換装置のソフトウェアとなる場合には、請求項10のように、かかるソフトウェアを記録した記録媒体上においても当然に存在し、利用されるといわざるをえない。むしろ、その記録媒体は、磁気記録媒体であってもよいし光磁気記録媒体であってもよいし、今後開発されるいかなる記録媒体においても全く同様に考えることができる。また、一次複製品、二次複製品などの複製段階については全く問う余地無く同等である。その他、ソフトウェアである場合にインストール作業において上述したような格子点を増加させる処理を行うことも可能であるし、供給方法として通信回線を利用して行なう場合でも本発明が利用されていることにはかわりないし、提供する側はソフトウェア提供装置として機能するものであり、同様に本発明を利用していることに相違ない。

【0033】さらに、一部がソフトウェアであって、一部がハードウェアで実現されている場合においても発明の思想において全く異なるものではなく、一部を記録媒体上に記憶しておいて必要に応じて適宜読み込まれるような形態のものとしてあってもよい。さらには、かかる色変換テーブルを使用することになるカラーファクシミリ機やカラーコピー機においても適用可能であることはいうまでもない。

10

20

30

40

50

【0034】

【発明の効果】以上説明したように本発明は、小さなサイズの元色変換テーブルから格子点を増した色変換テーブルを生成するものであるため、色変換をしない状態では最低限の記憶資源しか必要としない一方、色変換を行うときには必要な大きさとなった色変換テーブルとすることができ、より融通性の高い色変換テーブルの製造装置を提供することができる。むしろ、必要なときにだけ展開して不要ときには展開しないようにしても良いし、記憶資源に余裕があるのであれば展開した状態のまま残しておくこともできる。

【0035】また、請求項2にかかる発明によれば、非線形補間演算で格子点を増加させるようにしているため、増加された格子点の精度が高くなり、良好な色変換結果を得ることができる。これは、逆にいえばより格子点を少なくとも良好な結果を得ることができるという効果にもなる。

【0036】さらに、請求項3にかかる発明によれば、元色変換テーブルの格子点を均等な間隔とすることにより非線形補間演算が複雑化しないようにでき、演算時間などの減少を図ることができる。

【0037】さらに、請求項4にかかる発明によれば、演算量の少ない線形補間を利用して簡易に補間演算を行うことができる。また、線形補間の簡易さを利用すれば大きな変化部分で格子点を密にすることにより、簡易でありながら精度の向上を図ることができるなどのメリットもある。

【0038】さらに、請求項5にかかる発明によれば、増加させる格子点の数を選択できるので、ユーザの記憶資源に合わせて色変換テーブルのサイズを決めるなど、より柔軟にユーザの環境に適した色変換テーブルを生成することができるようになる。これはまた、複数のサイズの色変換テーブルを生成することもできるので、必要に応じて適宜選択して利用するといったことも可能となる。

【0039】さらに、請求項6にかかる発明によれば、環境に応じた数の格子点を選択するので、ユーザが煩わしい設定作業を行わなくても良くなる。

【0040】さらに、請求項7にかかる発明によれば、格子点の増加を変換画像に応じて選択するようにしているため、不要に色変換テーブルを大きくしすぎたり、不十分にしか色変換テーブルを大きくしていないなどといった不具合を無くすることができる。

【0041】さらに、請求項9にかかる発明によれば、上述したのと同様に、より融通性の高い色変換テーブル生成することが可能な色変換テーブルの製造方法を提供することができる。

【0042】さらに、請求項19にかかる発明によれば、上述したのと同様に、より融通性の高い色変換テーブル生成することが可能な色変換テーブルの製造プログ

ラムを、コンピュータ等で読み取り可能に記録した記録媒体として提供することができる。

【0043】

【発明の実施の形態】以下、図面にもとづいて本発明の実施形態を説明する。

【0044】図1は、本発明の一実施形態にかかる画像処理システムをブロック図により示しており、図2は具体的ハードウェア構成例をブロック図により示している。

【0045】同図において、画像入力装置10はカラー画像を撮像するなどして階調表色データを画像処理装置20へ出力し、同画像処理装置20は所定の画像処理を行なって画像出力装置30に出力し、同画像出力装置30は元のカラー画像を表示する。

【0046】ここにおいて、画像入力装置10の具体例はスキャナ11やデジタルスチルカメラ12などが該当し、画像処理装置20の具体例はコンピュータ21とハードディスク22などからなるコンピュータシステムが該当し、画像出力装置30の具体例はプリンタ31やディスプレイ32等が該当する。また、本発明をコンピュータ等に実施させるプログラムを記録可能な記録媒体は、ドライブ装置23によってコンピュータに読み込まれるCD-ROM24等の記録媒体が相当する。

【0047】画像入力装置10としてのスキャナ11が階調表色データとして例えばRGB（緑、青、赤）の階調データを出力するものとするとともに、画像出力装置30としてのプリンタ31は階調表色データとしてCMY（シアン、マゼンダ、イエロー）の二値データを入力として必要とするものとする、画像処理装置20としてのこのコンピュータ21の具体的役割は、RGBの階調データをCMYの二値データに変換することである。また、ディスプレイ32がRGBの階調データを入力するものとしても、スキャナ11とディスプレイ32では色特性が異なるのが通常であり、コンピュータ21はRGBの階調データをRGBの階調データに変換する処理を行なうことになる。デジタルスチルカメラ12についてもほぼ同様のことがいえる。

【0048】このコンピュータ21の内部で行なわれる処理を図3に示している。図に示すように、コンピュータ21内ではオペレーティングシステム21aが稼働しており、プリンタ31やディスプレイ32に対応したプリンタドライバ21bやビデオドライバ21cが組み込まれている。一方、アプリケーション21dはオペレーティングシステム21aにて処理の実行を制御され、必要に応じてプリンタドライバ21bやビデオドライバ21cと連携して所定の処理を実行する。

【0049】アプリケーション21dで生成される印刷用データはオペレーティングシステム21aを介してプリンタドライバ21bに入力され、当該プリンタドライバ21bはプリンタ31が要求するフォーマットの画像

データに変換する。この変換が上述したRGBの階調データをCMYの二値データに変換する処理に該当する。ここにおいて、同プリンタドライバ21bは、アプリケーション21dが所定の画面単位で生成する画像データからプリンタ31における印刷ヘッドの走査範囲を切り出すラスライザ21b1と、この走査範囲の各画素について色変換テーブルを参照してRGBの階調データをCMYの階調データに変換する色変換部21b2と、CMYの階調データを二値データに階調変換する階調変換部21b3とから構成されている。なお、アプリケーション21dが生成する表示画像データについてはビデオドライバ21cが所定の画面用メモリに書き込み、ハードウェア回路を介してディスプレイ32にて表示させている。

【0050】色変換部21b2は色補正モジュールとも呼ばれ、色変換の演算処理を実行する色変換用ソフトウェア21b2aと色変換テーブル21b2bとから構成されている。色変換テーブル21b2bは異なる表色空間の間で階調表色データを変換するために変換元の表色空間での格子点に変換先の表色空間での階調表色データを対応させたものであり、より具体的には図4に示すような三次元のRGB階調データを座標値としてCMY階調データを読み出すための三次元ルックアップテーブルである。そして、色変換用ソフトウェア21b2aは各画素のRGB階調データを座標値としてCMY階調データを読み出す処理を行う。

【0051】この色変換部21b2を含めて同プリンタドライバ21bは、図5に示すインストールプログラムによってハードディスク22上に展開される。このインストーラは、機器チェックを行なうステップS110と、上記色変換用ソフトウェア21b2aを含むドライバ用ソフトウェアをハードディスク22上に展開するステップS120と、所定の補間演算によって小サイズの*

$$P(X) = \sum_{i=0}^{N-1} \{ Y_i \prod_{j \neq i} ((X - X_j) / (X_i - X_j)) \}$$

【0059】がラグランジュ(Lagrange)の補間公式である。なお、右辺の \prod 以下については $((X - X_j) / (X_i - X_j))$ を $j = i$ 以外の全ての j について掛け合わせたものを意味する。この補間演算の具体的な実行方法をC言語で示したコーディングリストを図6に示している。

【0060】さて、このような補間演算を用いれば、図7に示すような実際には各軸方向に五つの格子座標しか持たない小サイズの元色変換テーブル21b2cを使用して各格子座標間に新たな三つの格子座標を増加して格子座標間を四分することは可能である。この場合の格子座標間隔は「64」階調に相当し、元色変換テーブル21b2cの各格子座標に格子番号「0」～「4」を付すものとする。また、新たに格子座標間を四分することに※50

*元色変換テーブル21b2cから所定のサイズの色変換テーブル21b2bを生成するステップS130とから構成されている。

【0052】すなわち、補間演算によって小サイズの元色変換テーブル21b2cから所定のサイズの色変換テーブル21b2bを生成するステップS130こそが本発明における色変換テーブルの製造装置を構成し、その手順が色変換テーブルの製造方法を構成している。この具体的手法については後述するとして、本実施形態においては、プリンタドライバ21bのインストーラとして具現化されているものの、その機能として小サイズの元色変換テーブル21b2cから所定のサイズの色変換テーブル21b2bを生成するものであればよい。従って、色変換テーブルを単独で生成するソフトウェアであってもよいし、あるいはワイヤーロジックからなるハードウェアなどで構成することも可能である。また、後述するように色変換用ソフトウェア21b2aが必要に応じて色変換テーブル21b2bを生成するよう構成することも可能である。

【0053】次に、この補間演算処理について詳述する。

【0054】まず、補間演算処理の一例として非線形補間演算を採用する場合について説明する。

【0055】 n 個の点 (X_i, Y_i) ($i = 0, 1, \dots, n-1$) が与えられれば $Y_i = P(X_i)$ ($i = 0, 1, \dots, n-1$) を満たす $n-1$ 次の多項式

【0056】

【数1】

$$P(X) = c_{n-1} \cdot X^{n-1} + c_{n-2} \cdot X^{n-2} + \dots$$

【0057】が一意的に定まる。但し、どの二つの X_i も等しくないとする。この多項式を表す閉じた式

【0058】

【数2】

※より色変換テーブル21b2bでは格子番号「0」～「16」となり、格子座標間隔は「16」階調になる。なお、本来「256」階調としてこのように均等に分配する格子点を設けるのは不可能であり、計算上は「0」～「256」というような(実際の階調数+1)の格子番号を想定しておき、計算の最後で最後の格子番号(例えば256)を現実の階調の範囲の最後の格子番号(255に相当)に移行させることによって計算を簡易にする。

【0061】元色変換テーブル21b2cのインストールプログラム内での記憶フォーマットについて概略を触れておくと、図8の上段に示すように、RGBの各成分を座標値としてそれぞれCMYの三色のデータを対応すべく、要素数が(5, 5, 5, 3)の配列となってお

11

り、ファイルの先頭からベタで書き込まれている。従って、元色変換テーブル21b2cの対応データを参照するためにはR軸とG軸とB軸とのそれぞれに対応した格子番号ポインタPr, Pg, Pbを設定し、ファイルの先頭から $(Pr \times 5 \times 5 \times 3 + Pg \times 5 \times 3 + Pb \times 3)$ をオフセットアドレスとしてシアン(C)は「1」バイト目、マゼンダ(M)は「2」バイト目、イエロー(Y)は「3」バイト目を読み出すことになる。

【0062】むしろ、かかる記憶フォーマットは一例に過ぎず、例えば、シアンについて全座標分だけ並べ、マゼンダ、イエローについて、順次、全座標分だけ並べていくなどの配置でも構わない。あるいは、ファイル圧縮した状態で保存してあっても良い。ただし、配列としてベタで書き込まれている場合には、後述するようにポインタ値で読み出しアドレスを演算でき、読み出す際の規則性を自由に設定できる。

【0063】一方、色変換テーブル21b2bの記憶フォーマットを図8の下段に示している、上述したように元色変換テーブル21b2cの各格子点間を四分して新たに三つの格子座標を形成するものであるため、要素数が(17, 17, 17, 3)の配列となっており、ファイルの先頭からベタで書き込むようにしている。従って、この色変換テーブル21b2bの対応データを参照するためには先程と同様の格子番号ポインタPr, Pg, Pbを設定すると、ファイルの先頭から $(Pr \times 17 \times 17 \times 3 + Pg \times 17 \times 3 + Pb \times 3)$ をオフセットアドレスとしてシアン(C)は「1」バイト目、マゼンダ(M)は「2」バイト目、イエロー(Y)は「3」バイト目を読み出すことになる。

【0064】色変換テーブル21b2bの格子点を増加する処理を図9のフローチャートに示している。この処理では、図8に示すように元色変換テーブル21b2cと色変換テーブル21b2bとで一致する格子点が存在するので、ステップS210にて格子点データを移転させる作業を行いつつ、格子点の間に空白のデータを挿入してハードディスク22上にファイルの形として展開する。この後、ステップS220にて色変換テーブル21b2bの全格子点について対応データを埋めるべく、各軸の格子番号についてネストしたループ処理を実行する。格子番号は「0」～「16」であるので、R軸、G軸、B軸についてポインタに「0」～「16」を設定して処理を繰り返す。最も内側のループ内ではR軸、G軸、B軸のポインタで示される格子点が元色変換テーブル21b2cから移行した格子点に一致するか否かを判断し、一致しなければ補間演算で格子点の対応データを算出する処理を実行する。しかし、一致するのであれば既に対応データがあるので補間演算の処理をスキップする。

【0065】補間演算の処理の一例を、図10～図12に示している。まず、フローを説明する前に図11にて

12

図示した非線形演算の概念を説明する。

【0066】数2に示したラグランジュの補間公式を四点の対応データに基づいて適用しようとした場合、図11に示すP点(Rp, Gp, Bp)の補間演算を行なうこととしても、必ずしも四つの格子点を通過するかどうかは不明である。従って、P点が位置する前後で各軸方向に四つの格子点からなる立方体を想定し、この立方体内で各軸方向ごとに順に補間演算を実行することにより、P点の演算に必要な四つの点の対応データを算出していくことにする。ここにおいて、各軸毎の格子座標を{R1, R2, R3, R4} {G1, G2, G3, G4} {B1, B2, B3, B4}と設定しておく。

【0067】まず、P点(図示△の点)を通過するG軸方向に平行な直線を想定すると、この直線は、G軸の格子座標を通過することになる四つのRB平面を貫通することになる。この各交点は同図にて○点で示しており、その座標は(Rp, G4, Bp)、(Rp, G3, Bp)、(Rp, G2, Bp)、(Rp, G1, Bp)である。この交点自体の対応データは不明であるため、それぞれの交点と交わるRB平面上でB軸に平行な直線を想定する。この直線はB軸の格子座標を通過することになる四つのRG平面を貫通する。四つの直線のうちG軸の座標が「G1」である点に注目し、各交点を同図にて●点で示している。その座標は(Rp, G1, B1)、(Rp, G1, B2)、(Rp, G1, B3)、(Rp, G1, B4)であり、まだ対応データは不明である。しかしながら、これらの交点を通過するR軸に平行な直線を想定すると、今度は全て格子点を通過する。すなわち、交点(Rp, G1, B1)を通過する直線は(R1, G1, B1)、(R2, G1, B1)、(R3, G1, B1)、(R4, G1, B1)を通過する。

【0068】これを逆に遡ることにすれば、四つの(R1, G1, B1)、(R2, G1, B1)、(R3, G1, B1)、(R4, G1, B1)から一つの●点の対応データを得ることができることになり、同様にして四つの●点の対応データを得たときには一つの○点の対応データを得ることができる。これを繰り返せば四つの○点の対応データを得ることができ、そうなれば△点の対応データを算出できるようになる。

【0069】かかる過程のより具体的な演算を図12に示しており、一番内側のネストでは $i = 1 \sim 4$ とした四つの格子点の対応データD(Ri, Gj, Bk)を利用し、R軸方向での成分値Rpでの対応データf(j)(●点の対応データ)を算出している。j = 1～4として四つのf(j)が得られれば、一つ上のネスト内ではこれを利用してB軸方向での成分値Bpでの対応データg(k)(○点の対応データ)を算出する。そして、k = 1～4として四つのg(k)が得られれば、一番上のネスト内ではこれを利用してh(△点の対応データ)が算出できる。

【0070】図10に示すフローチャートに戻ると、ステップS310では所属格子グループの特定を実行する。図11及び図12に示すように各軸方向に四つの格子点を固定して演算を実行すると容易であるため、演算のルーチンをこの立方体の座標値を利用して実行できるサブルーチン化している。従って、格子点を補間する演算を実行する前に当該格子点を含むような各軸方向に四つの格子点からなる立方体を特定する。そして、ステップS320ではこの立方体の格子点における対応データを同ワークエリアへ移動させる。

【0071】ワークエリアでは図11に示す関係が特定されるため、続くステップS330では図12に示すネスト処理で非線形演算を実行する。なお、ワークエリアへ移動させる際には各軸方向へのオフセットが生じるため、移動させる際にオフセット量を保存し、増加する格子点についてもその座標値に同オフセット量を考慮した座標値(Rp, Gp, Bp)で計算する。なお、図12においては三次元での補間に対応して三段階のネストの処理となっているが、さらに高次元での補間に対応してネスト処理することも可能である。

【0072】このようにして元色変換テーブル21b2cの格子点以外の格子点で非線形演算を施していくことにより、各軸のループ処理を終了したときに完全な色変換テーブル21b2bを得ることができる。

【0073】上述した実施形態においては、非線形演算の具体的な処理としてラグランジュの補間公式を利用しているが、他の演算を利用することも可能であり、例えば、スプライン(spline)補間も可能である。スプライン補間は導関数まで連続性を有する利用ができ、この意味で導関数の連続性が問題となる場合に備えた硬めの補間である。ただし、計算は複雑とならざるを得ず、このスプライン補間演算の具体的な実行方法をC言語で示したコーディングリストを図13に示している。

【0074】また、他の非線形の補間演算として、ネビル(Neville)補間であったり、ニュートン(Newton)補間などを利用可能である。これらの場合は、数値的にも計算が楽になる。

【0075】一方、上述した格子点については格子間隔を一定としている。これにより、特定の立方体をワークエリアとして対応データを移動させてしまい、演算を分かりやすく実行することができるようになる。しかしながら、必ずしも格子間隔が一定でなければならないわけではなく、格子間隔などを加味した係数などを利用して実行することも可能である。

【0076】さらに、上述した実施形態においては、ステップS220の補間演算処理で図10～図12に示す非線形補間を利用しているが、図14～図19に示すように線形補間を利用することも可能である。

【0077】図14には格子点を増加する前の格子点位置を白丸で示すとともに、格子点を増加した後の格子点

位置を黒丸で示しており、演算の簡易のため、格子間隔を半分にする位置に新たな格子点を設けている。従って、図に示す当初の格子番号は括弧書きのような丁度二倍の格子番号となる。なお、当初の格子点の数を仮に「i」として説明する。また、図15は線形補間の処理を実行するCPUの手順をフローチャートにより示しており、図16は最初の対応データの移動の状況を示しており、図17は補間される格子点を示しており、図18は補間演算の状況を示している。

- 10 【0078】格子点間を半分とする格子点を各軸に形成するものとする、補間前の格子点の格子座標は図14の括弧書きに示すように自動的に(0, 2, 4, 6, 8...)となり、その間を補間することになる。図15に示すフローチャートに戻ると、まず、CPUはステップS410にて既にテーブル内にある格子点データを新たなテーブルの所定位置に移行する処理を行う。例えば、図16に示すように、格子座標(0, 0, 0)の対応データは新たなテーブルの格子座標(0, 0, 0)の対応データとして、格子座標(0, 0, 1)の対応データは新たなテーブルの格子座標(0, 0, 2)の対応データとして、格子座標(0, 0, 2)の対応データは新たなテーブルの格子座標(0, 0, 4)の対応データとしてというようにして移行していく。
- 20 【0079】線形補間で格子点を補間する場合、周囲の八つの格子点からなる格子立法体内の位置によって演算が異なる。すなわち、辺上に存在する格子点の場合は両側の二点の格子点から補間されるし、面上に存在する格子点の場合は周囲の四つの格子点から補間されるし、中心に存在するもの場合は八つの格子点から補間される。

【0080】格子点を増加する順序として、まず、ステップS420では格子辺上で格子点を生成する処理を実行する。CPUの演算処理では各軸毎にパラメータを与えてネストしたループで処理を行うため、図中においてもブロックを入れ子状に表示している。

【0081】パラメータは各軸ともに「0」、「2」、「4」、「6」、「8」...と与え、R軸方向について例えば格子座標(1, 0, 0)の対応データを格子座標(0, 0, 0), (2, 0, 0)のデータから生成する。即ち、図18に示すように、格子座標(0, 0, 0)の対応データX1と格子座標(2, 0, 0)の対応データX2とを足し、その結果X3を「2」で割ったもののX4となる。ここにおいて「2」の除算は二進数データにおいて1ビットの右シフトに対応し、極めて容易に実行できる。むしろ、最初に1ビットの右シフトを実行しておいてから足しても良く、この場合は演算過程でのオーバーフローを防止できる。以下、このパラメータの全組合せから格子辺上の格子点を生成する。

【0082】ステップS430では格子面上で格子点を生成する処理を実行する。この場合もネストしたループ

で処理を行うため、各軸のパラメータとして「0」、「2」、「4」、「6」、「8」と与え、RG面と平行な面についていえば格子座標(1, 1, 0)の対応データを格子座標(0, 0, 0), (0, 2, 0), (2, 0, 0), (2, 2, 0)のデータから生成する。この場合は四つの格子点の平均値を取ることになり、四つのデータを足してから「4」で割ればよい。なお、「4」の除算は二進数データにおいて2ビットの右シフトに対応し、極めて容易に実行でき、以下、このパラメータの全組合せから格子面上の格子点は生成される。

【0083】最後に、ステップS440では中心点の格子点を生成する処理を実行する。この場合は、各軸のパラメータとして「1」、「3」、「5」、「7」…と与え、格子座標(1, 1, 1)の対応データは周縁の八つの格子座標(0, 0, 0), (0, 0, 2), (0, 2, 0), (0, 2, 2), (2, 0, 0), (2, 0, 2), (2, 2, 0), (2, 2, 2)の対応データから生成する。この場合は八つの格子点の平均値を取ることになり、オーバーフローしないように3ビットの右シフトを実行してから足し合わせればよい。以下、このパラメータの全組合せから全中心点の格子点が生成される。

【0084】以上の処理を行うことによって格子点の補間が終了する。本実施形態においては格子間隔を半分にするように格子点を増加させているが、この例に限らず、必要に応じて適宜増減可能であり、記憶資源の許容範囲内で格子点を増加させればよい。

【0085】また、上述した線形補間演算でも格子点の格子間隔が一定となっているが、線形補間においては格子間隔を適宜変えたとしても演算の基本は両側の二点の対応データにしか過ぎないため、格子間隔を変えたとしても演算は容易である。従って、対応データの変化カーブが大きい部分では格子間隔を細かくすることにより、演算容易のまま補間精度を向上させることができる。

【0086】一方、非線形補間においても線形補間においても、増加させる格子点を必ずしも一定とする必要はなく、必要に応じて変化させることも可能である。

【0087】格子点を増加させる処理の一例として、図19のフローチャートには、システムの機器構成に応じてテーブル生成するシステム対応格子点増加処理を示している。

【0088】この例では、ステップS510にて演算能力を表すCPUの種類を入力を行ない、ステップS520にて同様に演算速度を表すクロックの入力を行い、ステップS530にて演算能力や演算速度に影響するメモリ容量の入力を行い、ステップS540にて生成先のハードディスクの残り容量を入力する。

【0089】そして、これらの組合せに対応して予め設定されているシステム対応テーブルをステップS550にて参照し、最も適切な格子点の数を読み出す。格子点

の数を得られたらステップS560にて格子点増加処理を行う。このシステム対応テーブルに記憶されている格子点の数は、一般的な傾向として、演算能力や演算速度が速ければ格子の間隔が大きくなり、ハードディスクの残り容量が多ければ格子の間隔は小さくなるといったように設定しておけばよい。むろん、機器構成の入力要素はこれらに限るものではないし、その軽重も一定ではない。例えば、ハードディスクの残り容量が多い場合、キャッシュとの兼ね合いもあるもののフルサイズのテーブルを作成することも不可能ではない。

【0090】この場合、ユーザの使用環境として印刷する対象が写真などのビットマップ系のデータが多いかあるいはドローデータ系のデータが多いかをステップS540とステップS550との間で問い合わせるようにしても良い。そして、ビットマップ系のデータが多いならば、写真などの色再現性に重きを置かれている環境を想定して格子点の数を大きくすれば良いし、ドローデータ系のデータが多いならばビジネスグラフなどの色再現性があまり重要でない環境を想定して格子点の数を小さくすれば良い。

【0091】これまではインストール時に格子点の増加処理を行って色変換テーブル21b2bを生成するようにしているが、印刷時に必要なサイズの色変換テーブル21b2bをコンピュータ21のRAM等の高速アクセス可能な記憶部に生成するようにしてもよい。通常時は小さな元色変換テーブル21b2cをハードディスクに格納しておき、印刷実行時に必要なサイズの色変換テーブル21b2bを高速アクセス可能なRAM上に作るメモリは非常に大きい。むろんこの場合もハードディスクの場合と同様に、RAMでの利用可能な残り容量や、出力する画像データの量、望まれる出力品質等を参酌しながら格子の間隔を設定して展開するサイズを決めればよい。

【0092】図3に示すように、アプリケーション21dが印刷する場合にはオペレーティングシステム21aを介してプリンタドライバ21bが起動されるが、このときにファイルタイプがプリンタドライバ21bに渡される。プリンタドライバ21bではこの時のファイルタイプ(例えば、bmpなど)からビットマップ系であるのかドローデータ系であるのかを判断し、それに対応した格子点の数を設定して色変換テーブル21b2bを生成する。この格子点の数の大きさについては上述したインストーラによる場合と同様の傾向で設定すればよい。むろん、入力データの種類の判別する方法としてはこのようなファイルタイプだけに限らず、実際の入力データの色数が多いか少ないかなどによって判断しても良いし、オペレーティングシステムがオブジェクトの種類を判別してプリンタドライバに通知するようにしても良い。

【0093】このように、画像処理装置20を構成する

コンピュータ21にてインストーラが実行されると、ステップS130で元色変換テーブル21b2cから色変換テーブル21b2bを生成するが、このときの格子点増加処理では、ステップS430におけるラグランジュの補間公式を利用した非線形補間演算で格子点を増加させたり、線形補間で格子点を増加させるなどし、また、その際に固定した格子点の数であっても良いし環境や入力画像に応じた格子点の数としてもよく、小さなサイズの元色変換テーブル21b2cから適切なサイズの色変換テーブル21b2bを生成することができる。

【図面の簡単な説明】

【図1】本発明の一実施形態にかかる色変換テーブルの製造装置を適用した画像処理システムのブロック図である。

【図2】同画像処理システムの具体的ハードウェア構成例のブロック図である。

【図3】コンピュータのソフトウェア構成を示すブロック図である。

【図4】三次元ルックアップテーブルの概念を示す図である。

【図5】インストールプログラムのフローチャートである。

【図6】ラグランジュの補間演算をC言語でコーディングした図である。

【図7】元色変換テーブルの格子座標を示す図である。

【図8】元色変換テーブルと色変換テーブルのファイル構成を示す図である。

【図9】格子点増加処理プログラムのフローチャートである。

【図10】非線形補間プログラムのフローチャートである。

【図11】ラグランジュの補間公式で非線形補間する場合の手順を示す概念図である。

【図12】ラグランジュ補間演算に対応したフローチャートである。

【図13】スプライン補間演算をC言語でコーディングした図である。

【図14】格子点を増加する前後の格子座標を示す説明図である。

【図15】線形補間の格子点増加処理プログラムのフローチャートである。

【図16】元色変換テーブルと色変換テーブルのファイル構成を示す図である。

【図17】補間される格子点の位置を示す概略説明図である。

【図18】ビットシフトを併用した演算の状態を示す説明図である。

【図19】システム対応格子点増加処理プログラムのフローチャートである。

【符号の説明】

20…画像処理装置

21…コンピュータ

21a…オペレーティングシステム

21b…プリンタドライバ

21b1…ラスタイザ

21b2…色変換部

21b2a…色変換用ソフトウェア

21b2b…色変換テーブル

21b2c…元色変換テーブル

21b3…階調変換部

21c…ビデオドライバ

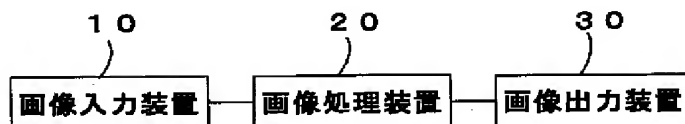
21d…アプリケーション

22…ハードディスク

23…ドライブ装置

24…CD-ROM

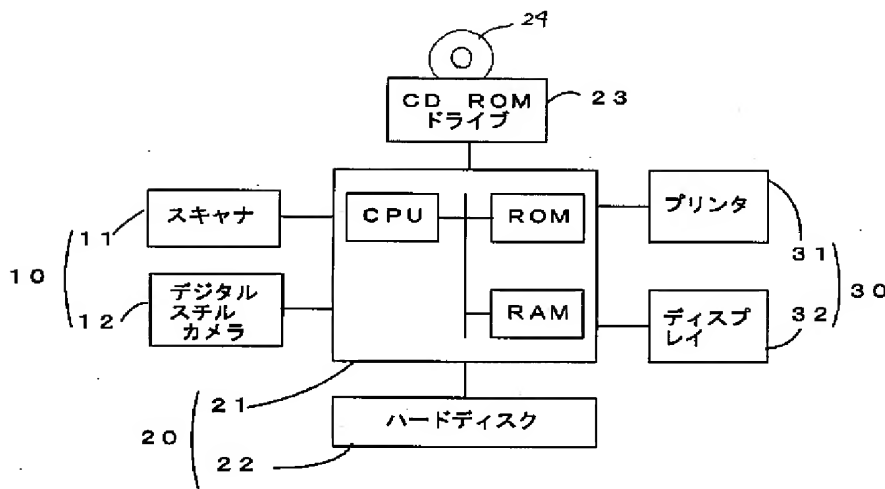
【図1】



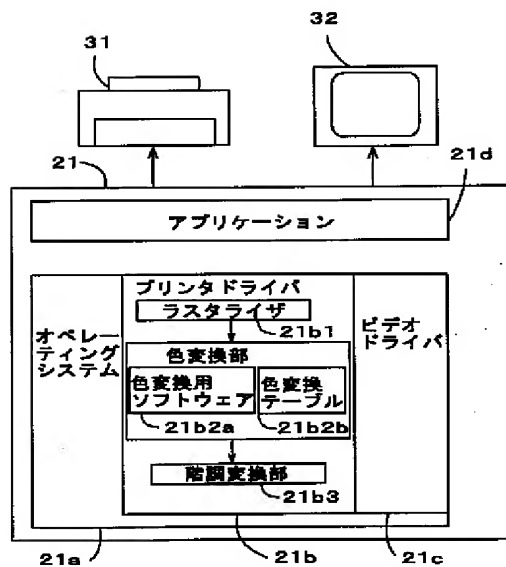
【図4】

$Cx(Rx, Gx, Bx)$
 $Mx(Rx, Gx, Bx)$
 $Yx(Rx, Gx, Bx)$

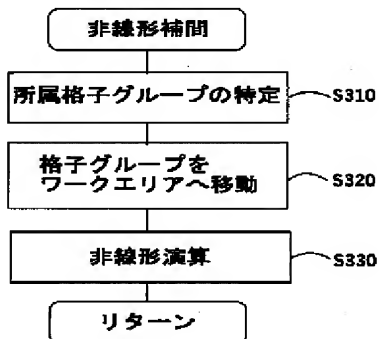
【図2】



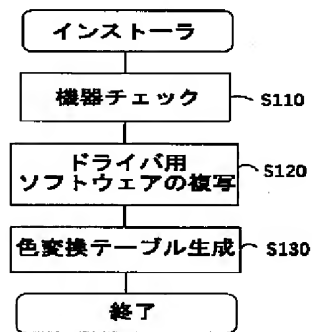
【図3】



【図10】



【図5】



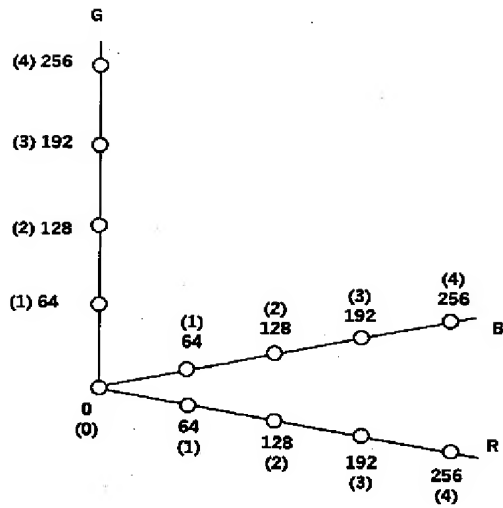
【図6】

```

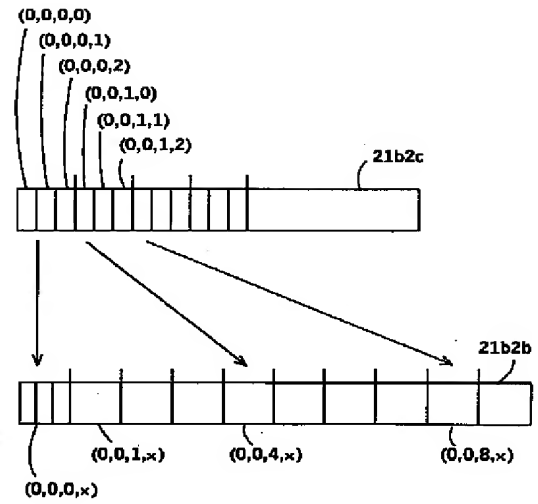
1 double lagrange(double t)
2 {
3     int i,j;
4     double sum,prod;
5
6     sum=0;
7     for (i=0;i<N;i++) {
8         prod=y[i];
9         for (j=0;j<N;j++)
10             if (j!=i) prod *= (t-x[j])/(x[i]-x[j]);
11         sum +=prod;
12     }
13     return sum;
14 }

```

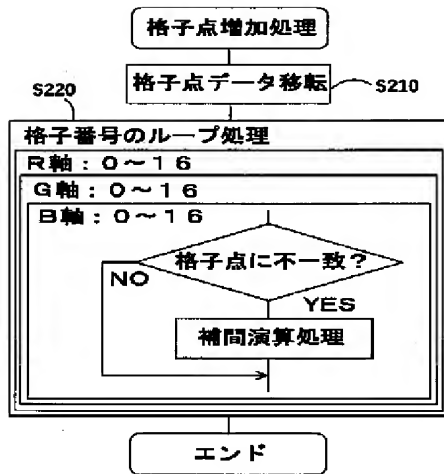
【図7】



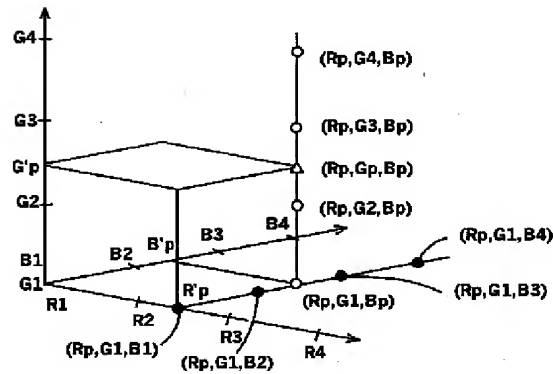
【図8】



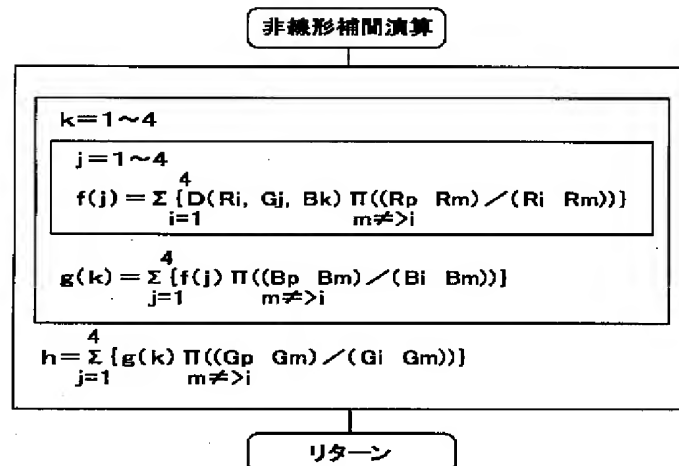
【図9】



【図11】



【図12】



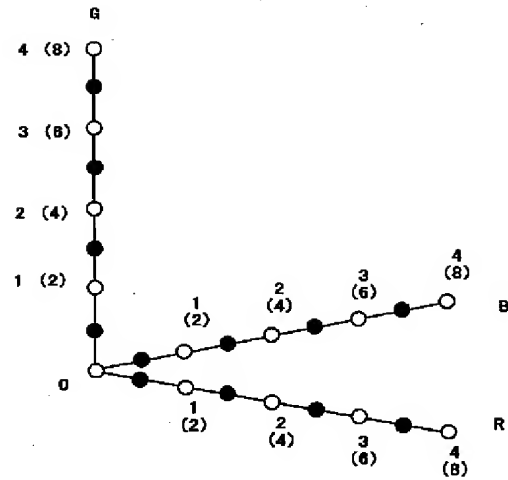
【図13】

```

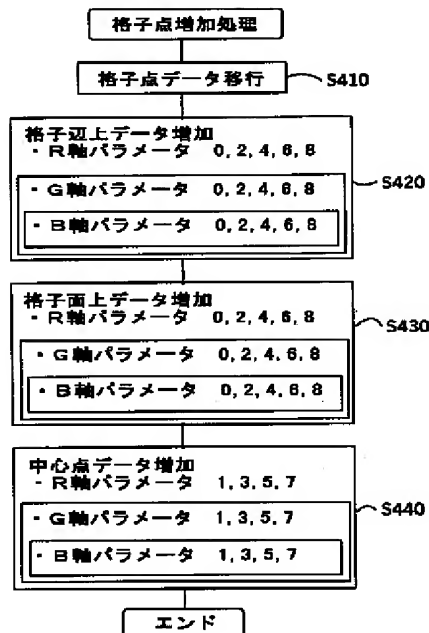
1 void maketable(double x[ ], double y[ ], double z[ ])
2 {
3     int i;
4     double t;
5     static double h[N], d[N];
6
7     z[0]=0; z[N-1]=0; /* 両端点での y N(x)/8 */
8     for (i=0; i < N-1; i++) {
9         h[i] = x[i+1]-x[i];
10        d[i+1]=(y[i+1]-y[i])/h[i];
11    }
12    z[1]=d[2]-d[1]-h[0] * z[0];
13    d[1]=2 * (n[2]-x[0]);
14    for(i=1; i < N-2; i++) {
15        t=h[i]/d[i];
16        z[i+1]=d[i+2]-d[i+1]-z[i] * t;
17        d[i+1]=2 * (x[i+2]-x[i])-h[i] * t;
18    }
19    z[N-2]=h[N-2] * z[N-1];
20    for(i=N-2; i > 0; i--)
21        z[i]=(z[i]-h[i] * z[i+1])/d[i];
22 }
23
24 double spline(double t, double x[ ], double y[ ], double z[
25 {
26     int i, j, k;
27     double d, h;
28
29     i=0; j=N-1;
30     while (i < j) {
31         k=(i+j)/2;
32         if(x[k] < t) i=k+1; else j=k;
33     }
34     if(i > 0) i--;
35     h=x[i+1]-x[i]; d=t-x[i];
36     return ((z[i+1]-z[i]) * d/h+z[i] * 3) * d
37            +((y[i+1]-y[i])/h
38            -(z[i] * 2 * z[i+1] * h)) * d+y[i];
39 }

```

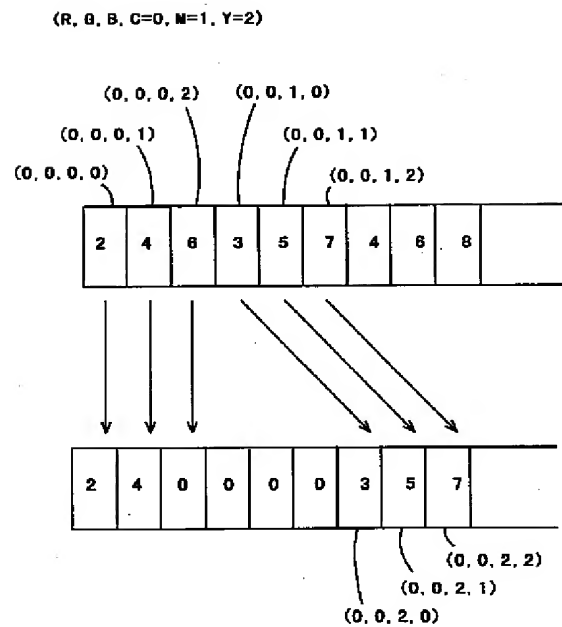
【図14】



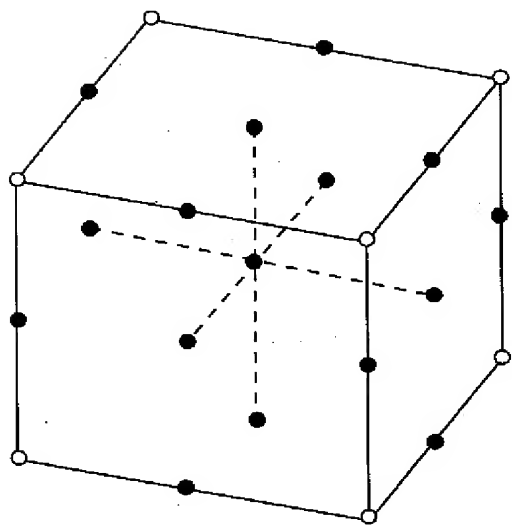
【図15】



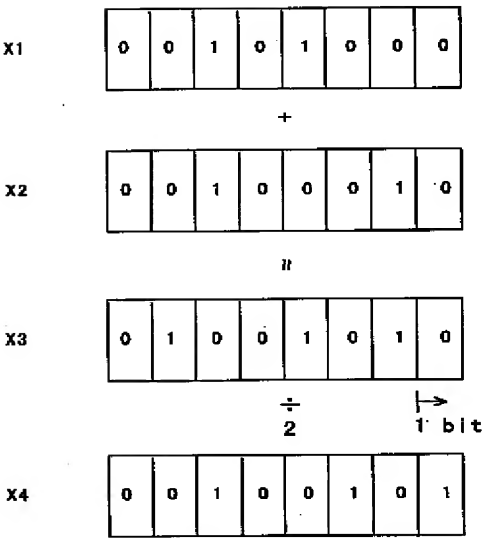
【図16】



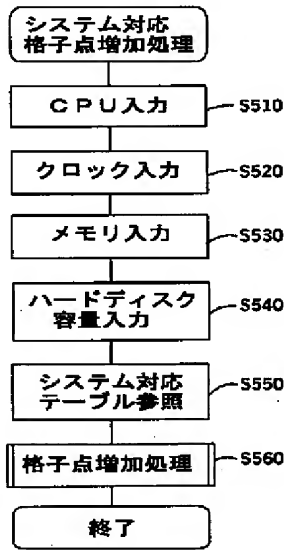
【図17】



【図18】



【図19】



フロントページの続き

(51)Int.Cl.⁶

識別記号

F I
H O 4 N 1/46

Z

PAT-NO: JP410191090A
DOCUMENT-IDENTIFIER: JP 10191090 A
TITLE: APPARATUS AND METHOD FOR
MANUFACTURING COLOR
CONVERSION TABLE, AND
RECORDING MEDIUM
PUBN-DATE: July 21, 1998

INVENTOR-INFORMATION:

NAME	COUNTRY
FUKAZAWA, KENJI	
KASAHARA, HIROKAZU	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
SEIKO EPSON CORP	N/A

APPL-NO: JP09274697
APPL-DATE: October 7, 1997

INT-CL (IPC): H04N001/60 , B41J002/525 ,
G06T001/00 , G09G005/06 ,
H04N001/46

ABSTRACT:

PROBLEM TO BE SOLVED: To generate a color conversion table optimum to the environment of the user or the like.

SOLUTION: When the installer is executed on a computer 2 being a component of an image processing unit, a color conversion table 21b2b is generated from an original color conversion table, number of grating points is increased by a nonlinear interpolation arithmetic operation or a linear interpolation arithmetic operation by utilizing the Lagrange's interpolation formula. In this case, number of the grating points may be fixed or number of grating points is selected in response to the environment or an input image and the color conversion table 21b2b with a proper size is generated from the original color conversion table with smaller size.

COPYRIGHT: (C)1998,JPO